REMARKS/ARGUMENTS

This Amendment and the following remarks are intended to fully respond to the Office Action dated February 26, 2004. In that Office Action, claims 1-58 were examined, and all claims were rejected. Reconsideration of these objections and rejections, as they might apply to the original and amended claims in view of these remarks, is respectfully requested.

Claims 1-58 are in the application. Claims 19, 31, and 55 have been amended and claims 35-37 have been canceled. No new claims have been added. Therefore, claims 1-34 and 38-58 remain present for examination.

Claim Objections

Claim 19 is objected to for an informality. The Applicant thanks the Examiner for pointing out this informality. Claim 19 has been amended as suggested. Therefore, the Applicant respectfully requests withdrawal of the objection.

Claim Rejections – 35 U.S.C. § 112

Claims 31-34 and 55-58 are rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claims 31 and 55, upon which 32-34 and 56-58 respectively depend, have been amended to more particularly point out and distinctly claim the subject matter which applicant regards as the invention. Therefore, the Applicant respectfully requests withdrawal of the rejection.

Claim Rejections – 35 U.S.C. § 101

Claims 35-37 are rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. Claims 35-37 have been canceled thereby rendering the rejection moot. Therefore, the Applicant respectfully requests withdrawal of the rejection.

Claim Rejections – 35 U.S.C. § 102

Claims 1-4, 6, 7, 9-13, 15, 17-23, 25-34, 38-41, 43, 44, 46-50, 52, and 54-58 are rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,189,142 of Johnston et al (hereinfter "Johnston"). The Applicant respectfully traverses the rejection and submits the following arguments pointing out significant differences between claims 1-4, 6, 7, 9-13, 15, 17-23, 25-34, 38-41, 43, 44, 46-50, 52, and 54-58 and Johnston.

Johnston relates to "providing runtime performance analysis in a visual programming environment." (col. 1, lines 18-19) Under Johnston, "providing runtime performance analysis in a computing system having a capability for visual programming, compris[es] instrumenting a selected one of one or more visually-created programs according to a visual programming performance data collection technique, and gathering execution information using the instrumented program." This instrumentation is achieved by "adding code hooks in appropriate locations within the code that implements the visual program." (col. 9, lines 49-51) That is, Johnston teaches adding code (i.e., "instrumentation") for debugging or tracing the execution of the visually created program. This instrumentation, for example, allows the developer to trace the flow of program execution while it is being tested. However, Johnston does not teach or suggest recognizing a history operator and a history operand in the source code as defined in the pending application.

As defined on page 10 of the description of the pending application, a history operand in source code represents a sequence of data associated with the history of an operand instance. Further, a history operator in source code is a shorthand representation of a function that object code will perform on the history data represented by the history operand. That is, history operators and operands are special, syntactical constructs that are recognized by the compiler or interpreter of the source code and trigger the compiler or interpreter to generate a particular object code. Examples of history operators and history operands are shown in Figs. 4-12 of the pending application. Therefore, by using history operands and operators such as described in the specification, a programmer has a shorthand way of specifying typical "bookkeeping" functions such as determining a minimum, maximum, mean, mode, etc. for a variable. See page 10 of the detailed description.

Independent claim 1, upon which claims 2-4, 6-7, 9-13, 15, and 17-18 depend, recites in part "recognizing a history operator and a history operand in the source code; generating first object code that, when executed, saves a data history associated with an instance of the history operand; and generating second object code that, when executed, performs the history operator on the data history." Johnston does not teach or suggest recognizing a history operator and a history operand in the source code, generating first object code that, when executed, saves a data history associated with an instance of the history operand, and generating second object code that, when executed, performs the history operator on the data history. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program. For at least these reasons, independent claim 1 and its dependent claims 2-4, 6-7, 9-13, 15, and 17-18 are distinguishable from Johnston and should be allowed.

Independent claim 19, upon which claims 20-23 and 25 depend, recites in part "a history operand to direct a translator to generate first object code that, when executed, saves a data history associated with an instance of the history operand; and a history operator to direct the translator to generate object second code that, when executed, performs the history operator on the data history." Johnston does not teach or suggest a history operand to direct a translator to generate first object code that, when executed, saves a data history associated with an instance of the history operand, and a history operator to direct the translator to generate object second code that, when executed, performs the history operator on the data history. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program. For at least these reasons, independent claim 19 and its dependent claims 20-23 and 25 are distinguishable from Johnston and should be allowed.

Independent claim 26, upon which claims 27-30 depend, recites in part "recognizing a history operand in source code; finding at least one instance of the history operand in the source code in response to recognizing the history operand; allocating storage; and generating first object code associated with each instance, wherein the first object code, when executed, saves a data history associated with the history operand in the storage." Johnston does not teach or suggest recognizing a history operand in source code, finding at least one instance of the history operand in the source code in response to recognizing the history operand, allocating storage, and generating first object code associated with each instance, wherein the first object code, when executed, saves a data history associated with the history operand in the storage. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program. For at least these reasons, independent claim 26 and its dependent claims 27-30 are distinguishable from Johnston and should be allowed.

Independent claim 31, upon which claims 32-34 depend, recites in part "recognizing a history operand in the source code, wherein the source code is contained in the memory; in response to recognizing the history operand, finding at least one instance of the history operand in the source code; allocating storage for a data history associated with the history operand; generating first object code associated with each instance, wherein the first object code, when executed, saves the data history associated with the history operand in the storage; and generating second object code that, when executed, performs the history operator on the data history." Johnston does not teach or suggest recognizing a history operand in the source code, wherein the source code is contained in the memory, in response to recognizing the history operand, finding at least one instance of the history operand in the source code, allocating storage for a data history associated with the history operand, generating first object code associated with each instance, wherein the first object code, when executed, saves the data history associated with the history operand in the storage, and generating second object code that, when executed, performs the history operator on the data history. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program. For at least these reasons, independent claim 31 and its dependent claims 32-34 are distinguishable from Johnston and should be allowed.

Independent claim 38, upon which claims 39-41, 43-44, 46-50, 52, and 54 depend, recites in part "recognizing a history operator and a history operand in the source code; saving a data history associated with an instance of the history operand; and performing the history operator on the data history." Johnston does not teach or suggest recognizing a history operator and a history operand in the source code, saving a data history associated with an instance of the history operand, and performing the history operator on the data history. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program. For at least these reasons, independent claim 38 and its dependent claims 39-41, 43-44, 46-50, 52, and 54 are distinguishable from Johnston and should be allowed.

Independent claim 55, upon which claims 56-58 depend, recites in part "recognizing a history operand in source code, the history operand representing a sequence of data associated with the history of an operand instance; finding at least one instance of the history operand in the source code in response to recognizing the history operand; and saving a data history associated with each instance of the history operand in the storage." Further, claim 56 recites in part "recognizing a history operator in the source code, the history operator representing a function that object code will perform on the data history associated with the history operand; and performing the history operator on the data history." Johnston does not teach or suggest recognizing a history operand in source code, the history operand representing a sequence of data associated with the history of an operand instance, finding at least one instance of the history operand in the source code in response to recognizing the history operand, and saving a data history associated with each instance of the history operand in the storage. Neither does Johnston teach or suggest recognizing a history operator in the source code, the history operator representing a function that object code will perform on the data history associated with the history operand and performing the history operator on the data history. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program. For at least these reasons, independent claim 55 and its dependent claims 56-58 are distinguishable from Johnston and should be allowed.

Claims 35-37 are rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,349,406 of Levine et al (hereinafter "Levine"). Claims 35-37 have been canceled thereby rendering the rejection moot. Therefore, the Applicant respectfully requests withdrawal of the rejection.

<u>**Claim Rejections – 35 U.S.C. § 103**</u>

Claims 5, 8, 14, 16, 24, 42, 45, 51, and 53 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Johnston in view of Levine. The Applicant respectfully traverses the rejection since the references, alone or in combination, fail to teach or suggest all the claimed limitation.

As discussed above, claim 1, upon which claims 5, 8, 14, and 16 depend, claim 19, upon which claim 24 depends, and claim 38, upon which claims 42, 45, 51, and 53 depend all relate to using a history operator and a history operand as described in the pending application. Johnston does not teach or suggest using a history operator and a history operand as described in the pending application. Rather, Johnston teaches adding code for debugging or tracing the execution of the visually created program.

Levine relates to "a method and system for compensating for instrumentation overhead in trace data by computing average minimum event times." (Abstract) More specifically, under Levine "in order to profile a program, the program is executed to generate trace records that are written to a trace file." (Col. 3, lines 16-18) The "trace data may be generated via selected events and timers through the instrumented interpreter without modifying the source code." (Col. 8, lines 14-16) The interpreter executes a trace program "used to record data upon the execution of a hook, which is a specialized piece of code at a specific location in a routine or program in which other routines may be connected." (Col. 9, lines 43-46) However, Levine does not teach or suggest using a history operator and a history operand as described in the pending application. Rather, Levine, similar to Johnston, teaches trace data generated by an interpreter upon execution of a hook code in the source program.

Independent claim 1, upon which claims 5, 8, 14, and 16 depend, recites in part "recognizing a history operator and a history operand in the source code; generating first object code that, when executed, saves a data history associated with an instance of the history operand; and generating second object code that, when executed, performs the history operator on the data history." Levine, either alone or in combination with Johnston, does not teach or suggest recognizing a history operator and a history operand in the source code, generating first object code that, when executed, saves a data history associated with an instance of the history operand, and generating second object code that, when executed, performs the history operator on the data history. For at least these reasons, independent claim 1 and its dependent claims 5, 8, 14, and 16 are distinguishable from the combination of Johnston and Levine and should be allowed.

Independent claim 19, upon which claim 24 depends, recites in part "a history operand to direct a translator to generate first object code that, when executed, saves a data history associated with an instance of the history operand; and a history operator to direct the translator to generate object second code that, when executed, performs the history operator on the data history." Johnston, either alone or in combination with Levine, does not teach or suggest a history operand to direct a translator to generate first object code that, when executed, saves a data history associated with an instance of the history operand, and a history operator to direct the translator to generate object second code that, when executed, performs the history operator on the data history. For at least these reasons, independent claim 19 and its dependent claim 24 are distinguishable from the combination of Johnston and Levine and should be allowed.

Independent claim 38, upon which claims 42, 45, 51, and 53 depend, recites in part "recognizing a history operator and a history operand in the source code; saving a data history associated with an instance of the history operand; and performing the history operator on the data history." Johnston, alone or in combination with Levine, does not teach or suggest recognizing a history operator and a history operand in the source code, saving a data history associated with an instance of the history operand, and performing the history operator on the data history. For at least these reasons, independent claim 38 and its dependent claims 42, 45, 51, and 53 are distinguishable from the combination of Johnston and Levine and should be allowed.

Conclusion

It is believed that no further fees are due with this Response. However, the Commissioner is hereby authorized to charge any deficiencies or credit any overpayment with respect to this patent application to deposit account number 13-2725.

In light of the above remarks and amendments it is believed that the application is now in condition for allowance. Applicants request the application be allowed and pass to issuance as soon as possible. Should any additional issues need to be resolved, the Examiner is requested to telephone the undersigned attorney to resolve those issues.

Respectfully submitted,

[illegible]

Dated: May 19, 2004

William J. Daley, Reg. No. 52,471
Merchant & Gould P.C.
PO Box 2903
Minneapolis, MN 55402-0903
303.357.1651

27488
PATENT TRADEMARK OFFICE